In [10]:
```
#importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

In [13]:
```
#loading the dataset
df=pd.read_csv('winequality-red.csv')
```

Out[13]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [14]:
```
df.info()
```

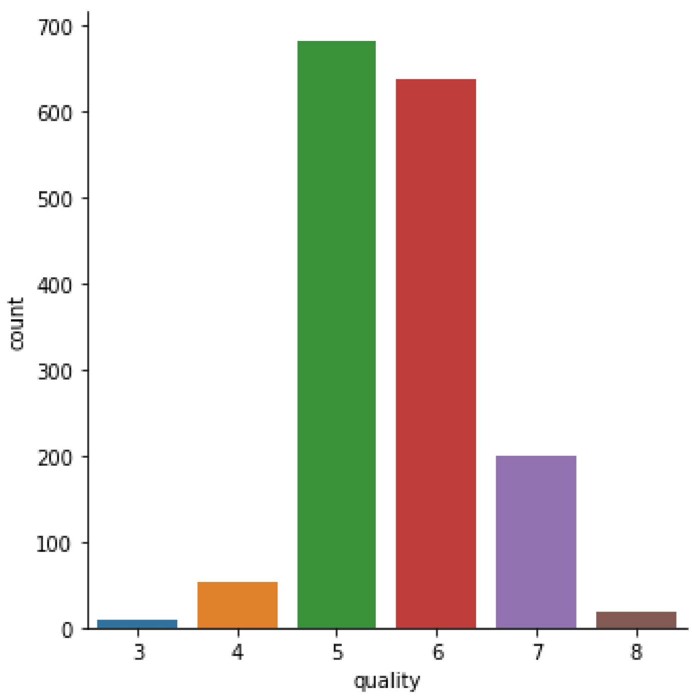
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [15]:
```
df.describe()
```

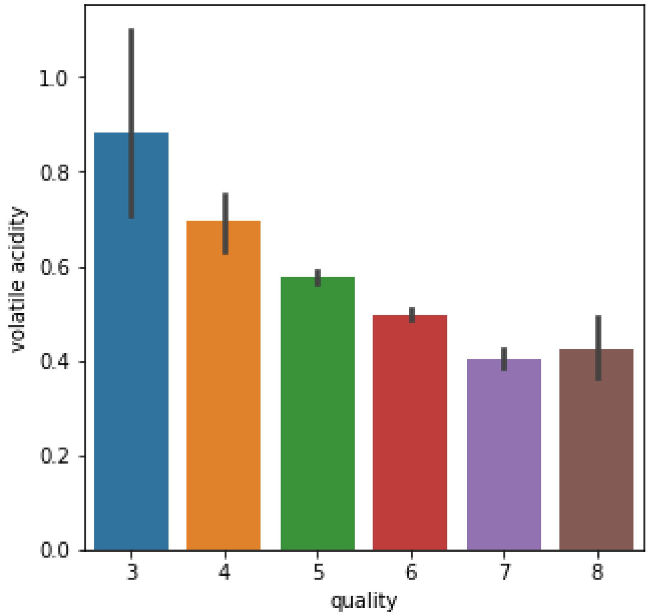
Out[15]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulf dioxi
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.4677
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.8953
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.0000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.0000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.0000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.0000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.0000

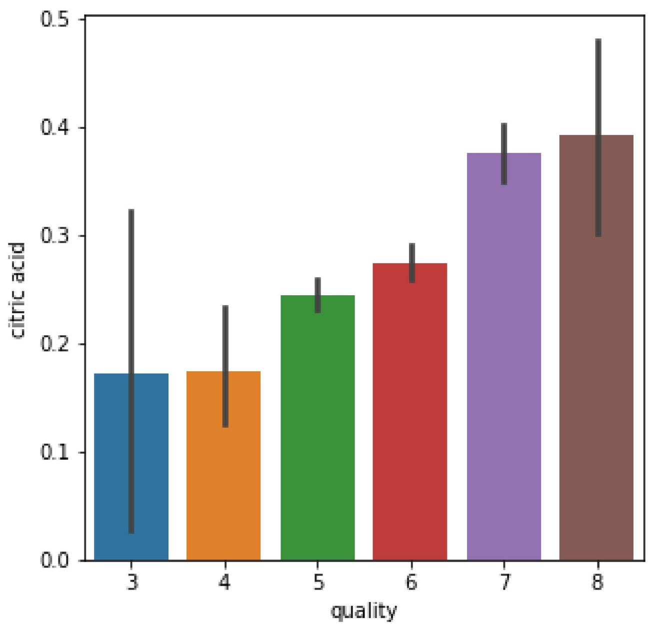
In [16]:
```
sns.catplot(x='quality',data=df,kind='count')
```



In [18]:
```
# volatile acidity vs Quality
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='volatile acidity',data=df)
plt.show()
```

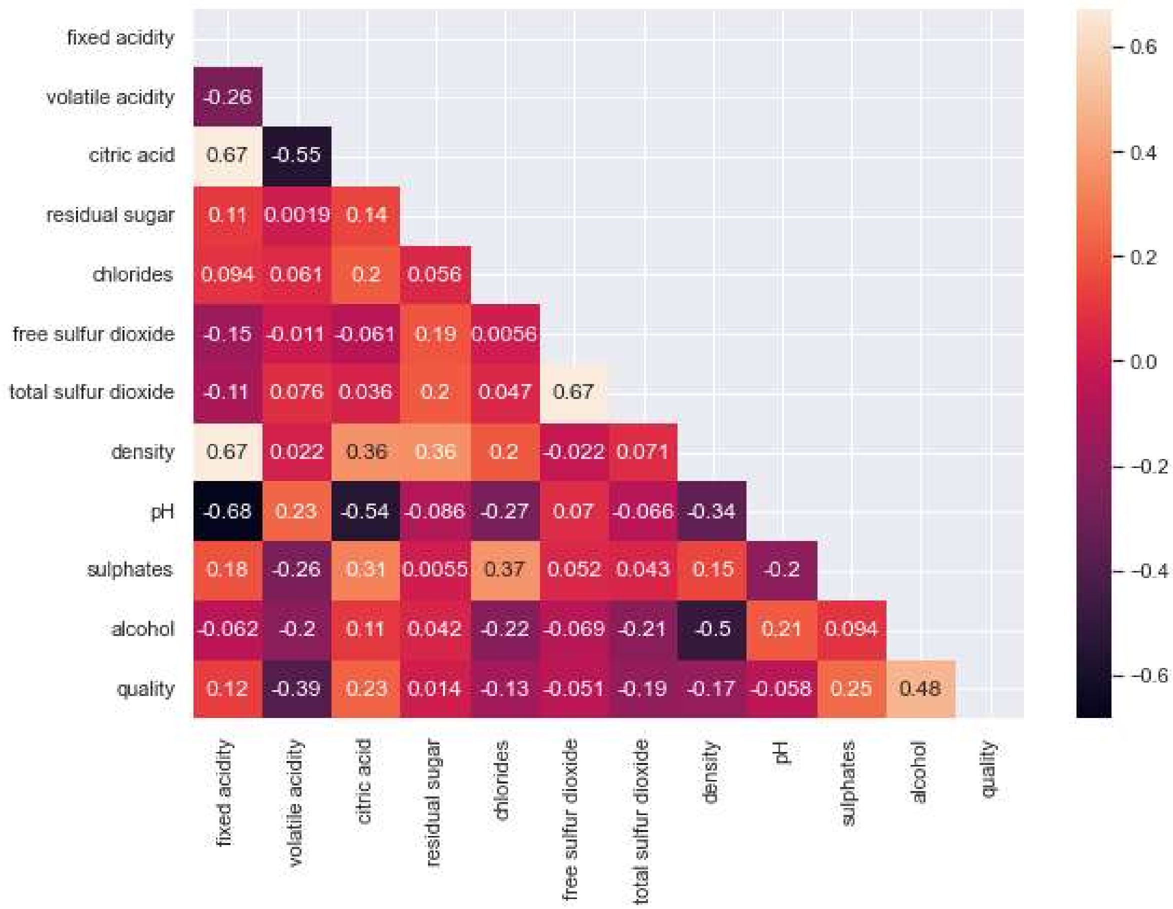


In [19]:
```
# citric acid vs Quality
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='citric acid',data=df)
```

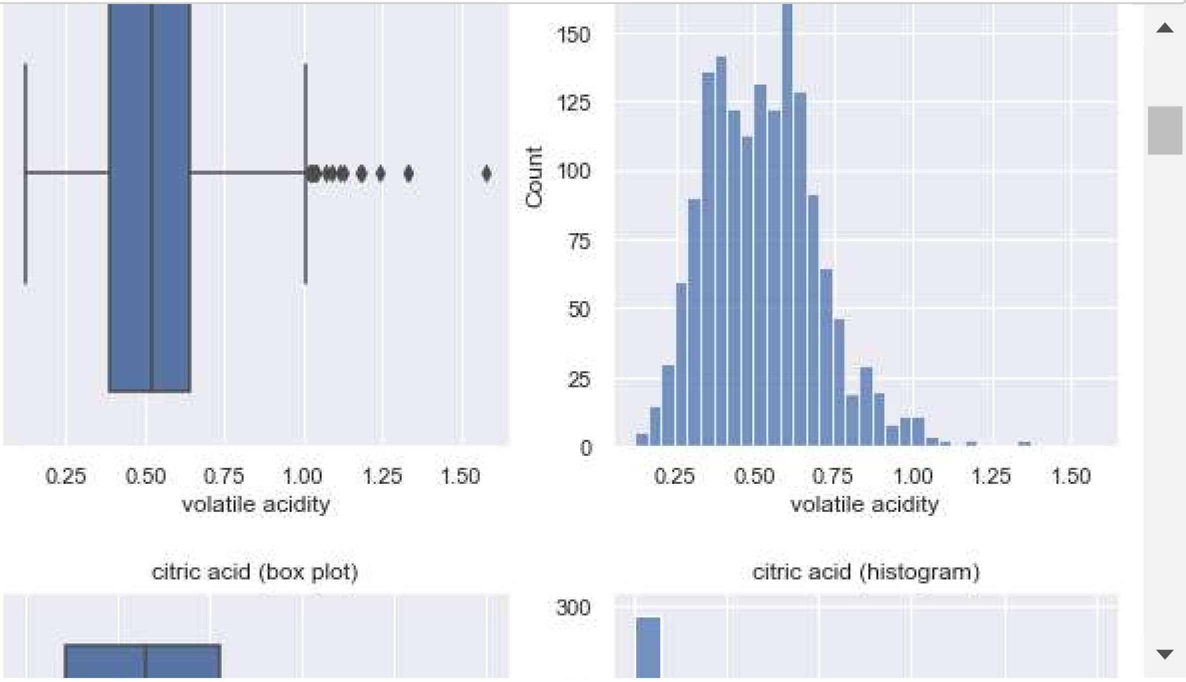


In [22]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 7))
mask = np.triu(df.corr())
```

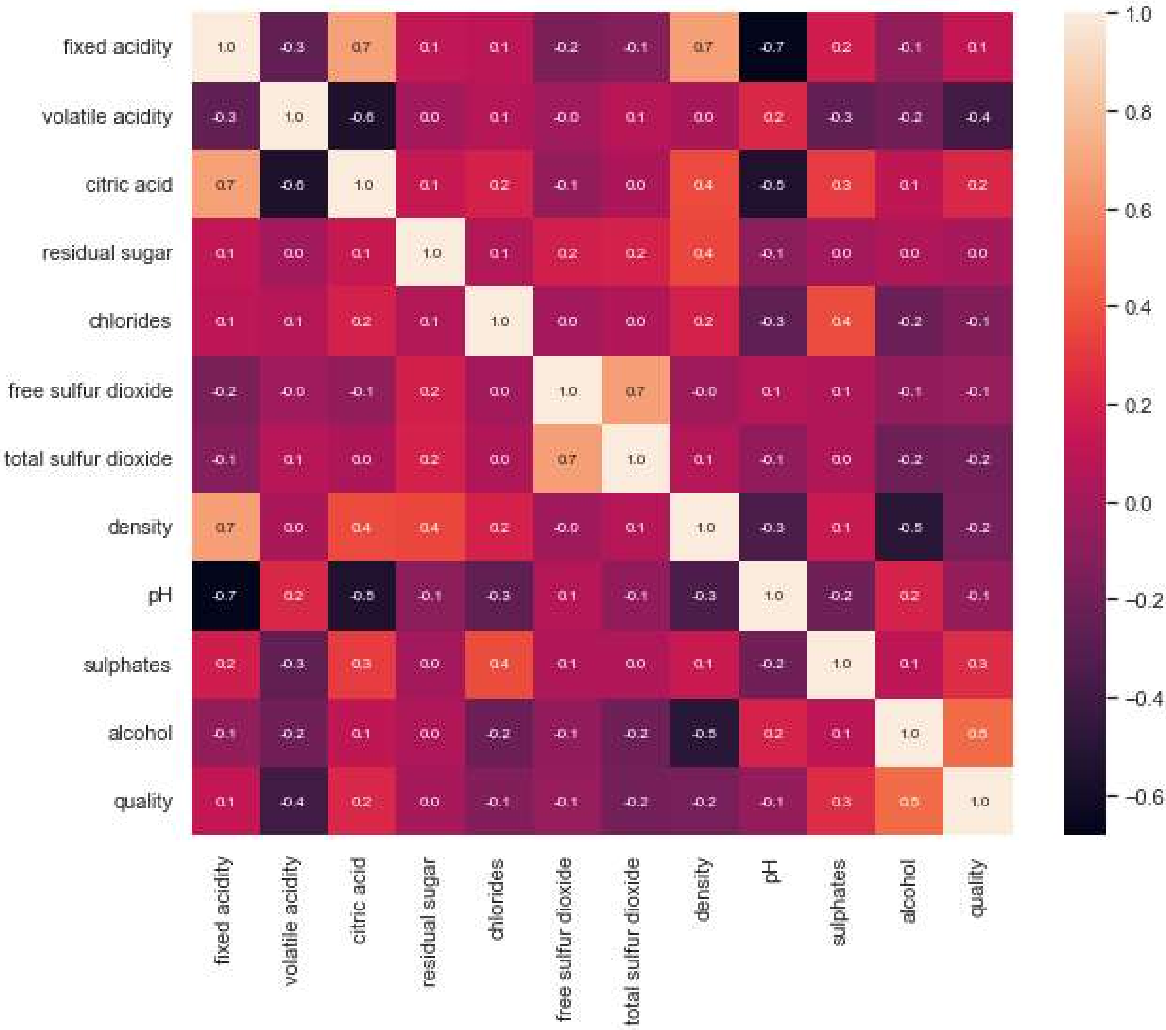
Out[22]: <Axes: >



In [24]:
```
for i in df.columns:
    fig, axs = plt.subplots(ncols=2, figsize=(10, 5))

    sns.boxplot(x=df[i], ax=axs[0])
    axs[0].set_title(i + ' (box plot)')

    sns.histplot(x=df[i], kde=False, ax=axs[1])
    axs[1].set_title(i + ' (histogram)')

    plt.show()
```



In [26]:
```
correlation=df.corr()
# constructing a heatmap to understand the correlation between the columns

plt.figure(figsize=(10,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={
```



In [27]:
```
features_df=df.drop('quality',axis=1)
```

Out[27]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	

1599 rows × 11 columns

In [28]:
```
target =df['quality'].apply(lambda y_value: 1 if y_value >=7 else 0)
```

Out[28]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: quality, Length: 1599, dtype: int64
```

In [30]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train , Y_test =  train_test_split(features_df,target,test_
print(f"Shape of X_train dataset: {X_train.shape}")
print(f"Shape of Y_train dataset: {Y_train.shape}")
print("*"*40)
print(f"Shape of X_test dataset: {X_test.shape}")
```

```
Shape of X_train dataset: (1279, 11)
Shape of Y_train dataset: (1279,)
****************************************
Shape of X_test dataset: (320, 11)
Shape of Y_test dataset: (320,)
```

In [31]:
```
from sklearn.ensemble._forest import RandomForestClassifier
model=RandomForestClassifier()
```

Out[31]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [33]:
```
# accuracy testing on test data
from sklearn.metrics import accuracy_score
X_test_pred=model.predict(X_test)
test_data_accuracy=accuracy_score(X_test_pred,Y_test)
```

```
Model Accuracy:  0.93125
```

In [39]:
```
df.sample(10)
```

Out[39]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
877	7.7	0.715	0.01	2.1	0.064	31.0	43.0	0.99371	3.41	0.57	
992	6.5	0.400	0.10	2.0	0.076	30.0	47.0	0.99554	3.36	0.48	
1171	7.1	0.590	0.00	2.2	0.078	26.0	44.0	0.99522	3.42	0.68	
484	10.6	0.440	0.68	4.1	0.114	6.0	24.0	0.99700	3.06	0.66	
842	10.6	0.500	0.45	2.6	0.119	34.0	68.0	0.99708	3.23	0.72	
1133	7.2	0.480	0.07	5.5	0.089	10.0	18.0	0.99684	3.37	0.68	
961	7.1	0.560	0.14	1.6	0.078	7.0	18.0	0.99592	3.27	0.62	
1390	6.0	0.490	0.00	2.3	0.068	15.0	33.0	0.99292	3.58	0.59	
1561	7.8	0.600	0.26	2.0	0.080	31.0	131.0	0.99622	3.21	0.52	
874	10.4	0.380	0.46	2.1	0.104	6.0	10.0	0.99664	3.12	0.65	

In [52]:
```
# Positive Result

input_data=(7.5,0.520,0.16,1.9,0.085,13.0,35.0,0.99680,3.38,0.62,9.5)

input_data_as_np=np.asarray(input_data)

input_data_reshaped=input_data_as_np.reshape(1,-1)

prediction=model.predict(input_data_reshaped)

print(prediction)

if(prediction[0]==1):
    print('Good Quality Wine')
else:
```

```
[1]
Good Quality Wine
```

In [55]:
```
# Negative Result

input_data=(7.3,0.305,0.39,1.2,0.058,7.0,13.0,0.99331,3.29,0.52,11.6)

input_data_as_np=np.asarray(input_data)

input_data_reshaped=input_data_as_np.reshape(1,-1)

prediction=model.predict(input_data_reshaped)

print(prediction[0])

if(prediction[0]==1):
    print('Good Quality Wine')
else:
```

```
0
Bad Quality Wine
```

Conclusion

Random Forest proves highly effective for Wine Quality Prediction, achieving a 93% accuracy on a test dataset. Key predictors are alcohol content, acidity, and residual sugar, emphasizing their crucial role in quality determination.

In [ ]:
```
```